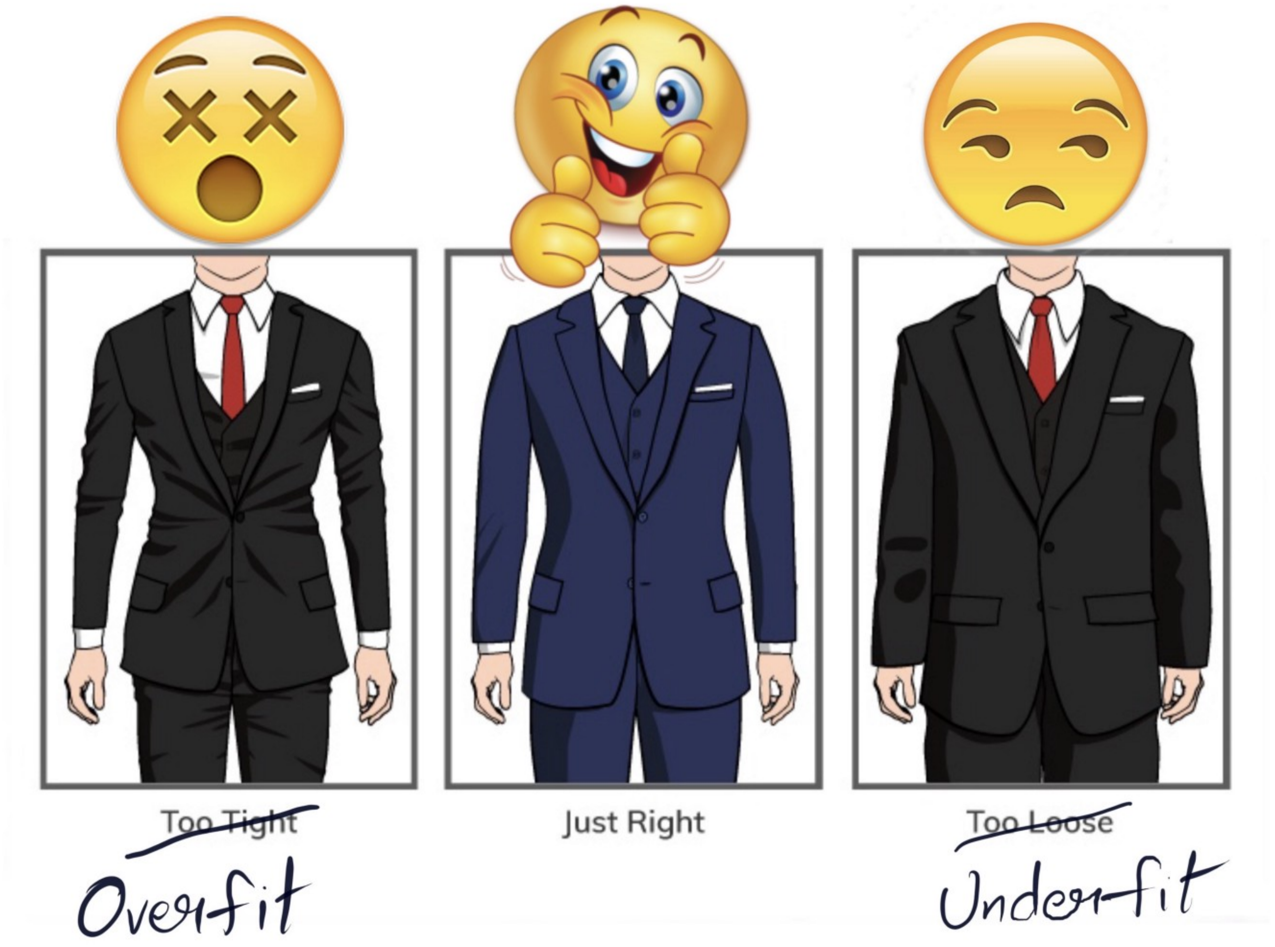
Regularization and tackling overfitting

Sowmya Yellapragada
Jan 27 · 7 min read

A cheatsheet to regularization in machine learning



Lest we forget, the goal of a machine learning is to learn from (training) data and build accurate predictive models that can find patterns and generalize on new and unseen, but similar data.

In the process of model building, we often encounter four scenarios:

1. **Low training error and low testing error**: This is the dream team.
2. **High training error and low testing error:** It can mean one of two things — either your model is psychic or you must have left a bug in your code. As a rule of thumb, "*It's always the latter*".
3. **High training error and high testing error**: This scenario is also known as *underfitting*. This means that the model is unable to generalize and learn from the training and thereby has low predictive power on the test dataset as well. We previously discussed this concept in one of our detailed discussions on bias and variance in machine learning

The tradeoff between Bias and variance

What is bias and variance trade-off? This is one of the first stepping stones in most ML interviews and preliminary ML…

medium.com

4. **Low training error and high testing error**: This scenario indicates that the model is able to learn well and holds good predictive power on the training dataset, however, perhaps it also learned the inherent noise in the training dataset, because of which it has low predictive power on the unseen data. This is the case of *overfitting*. In this article, we will focus on ways to combat this issue

Regularization

This is a set of strategies designed explicitly to reduce the test error, possibly at the expense of increased training error.

There are different kinds of regularization strategies applied to diverse machine learning models. We will briefly discuss these techniques for each model type in this section and elaborate some of them in much more detail in the following sub-sections

Linear models

- **L1 and L2 regularization** add a penalty to the cost function so that the model doesn't overfit on the training data. These are particularly useful in linear models i.e classifiers and regressors
- The resulting **cost function = loss + regularization term.**

Tree-based models

Regularization techniques for tree-based models include

- Limiting the maximum depth of trees,
- Setting stricter stopping criterion on when to split a node further (e.g. min gain, number of samples, etc.);
- Hyper-parameter tuning of the model and plot for train and cross-validation loss against hyper-parameters to identify the right fit to avoid overfitting
- Using ensemble models like the random forest in place of decision tree models.

Deep learning

- Deep learning is a subset of machine learning, and hence the regularization techniques like L1 and L2 are also applicable to these models.

However, there are more other regularization techniques that are catered to this area in particular like —

- Dropout
- Data augmentation
- Early stopping

L1 or LASSO regularization

- LASSO stands for — **L**east **A**bsolute **S**hrinkage and **S**election **O**perator

$$L_1(X,\omega) = L(X,\omega) + \lambda \sum |\omega_i|$$

$$\omega_i \leftarrow \omega_i - \eta \cdot [\partial L(X,\omega)/\partial\omega_i + \lambda \cdot sgn(\omega_i))]$$

The modified error function and weight gradient with L1 regularization

- Lasso penalizes the loss function by the sum of their absolute values (L1 penalty) of the coefficient vector, weighted by the regularization parameter λ
- This technique also results in feature selection, by zeroing out most of the non-relevant features. As a result, for high values of λ, many coefficients are exactly zeroed under lasso

L2 or Ridge regularization

$$L_2(X,\omega) = L(X,\omega) + \lambda \sum \omega_i^2$$

$$\omega_i \leftarrow \omega_i - \eta \cdot [\partial L(X,\omega)/\partial\omega_i + 2\lambda\omega_i]$$

The modified error function and weight gradient with L1 regularization

- Similar to the L1 regularization technique, L2 also introduces a regularization term to the loss function.
- The L2 parameter norm penalty is commonly known as weight decay, as it forces the weights to decay towards zero (but not exactly zero).

Choosing the value of the regularisation parameter (λ)?

- If the value of λ is too high, it will lead to extremely small values of the regression coefficient *w*, which will lead to the model underfitting (high bias — low variance).
- On the other hand, if the value of λ is 0 (very small), the model will tend to overfit the training data (low bias — high variance).
- There is no proper way to select the value of λ. What you can do is have a sub-sample of data and run the algorithm multiple times on different sets. Here, you need to decide how much variance can be tolerated. Once you are satisfied with the variance, that value of λ can be chosen for the full dataset. One thing to be noted is that the value of λ selected here was optimal for that subset, and might not be optimal for the entire training data.

Comparing L1 and L2 regularizations

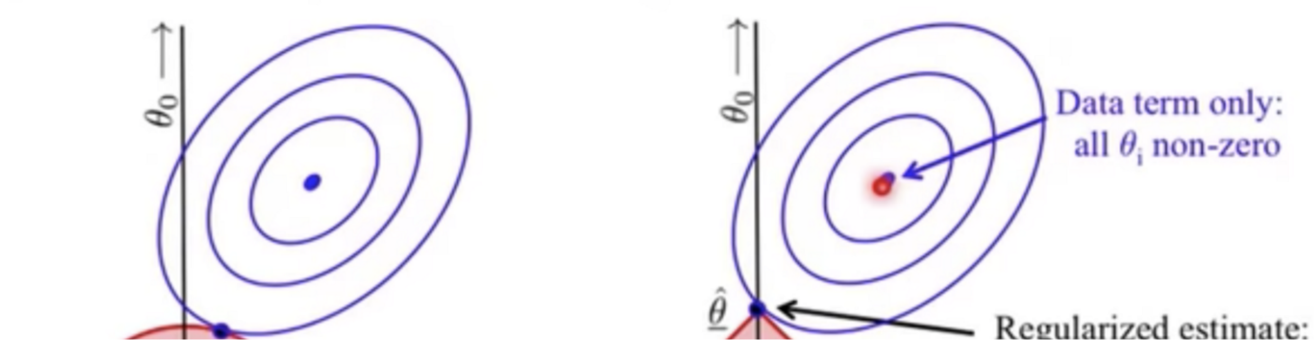
Robustness: L1 > L2

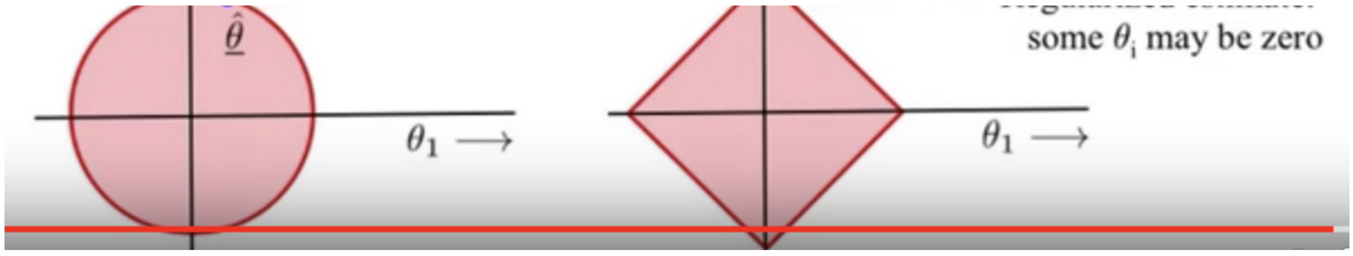
- The more able a model is to ignore extreme values in the data, the more robust it is.
- The L1 norm is more robust than the L2 norm, for fairly obvious reasons: the L2 norm squares values, so it increases the cost of outliers exponentially; the L1 norm only takes the absolute value, so it considers them linearly.

Stability: L2 > L1

- Stability is defined as resistance to horizontal adjustments. This is the perpendicular opposite of robustness.
- Suppose every weight coefficient is perturbed by a very small value Δ. Then the L1 norm will also be perturbed by a factor of Δ, whereas the change in L2 will be a factor of Δ^2 . Δ^2 is a very small value, hence the change in L2 is lesser than that in L1. Hence L2 is more resistant to perturbations

Sparse solutions : L1 > L2




some θ_i may be zero
$\hat{\theta}$
$\theta_1 \longrightarrow$
$\theta_1 \longrightarrow$

Credit: Alexander Ihler

- Here we plot the graphs for the optimization problem and the regularization functions — L1 and L2
- The blue curve here is the regular loss function. It is trying to optimize towards the true value at the center of the graph. Whereas, the red curves are the regularization terms which are optimizing towards the origin 0
- The modified loss function (with regularization), strives to strike a balance between these two and reach the optimal point theta
- The theta point in case of L1 regularization can be at 0, because of the structure of the L1 norm. However, this can happen with L2 only when the true label is also on the axis.
- Hence L1 norms encourage some level of sparsity. This makes models more efficient to store and compute

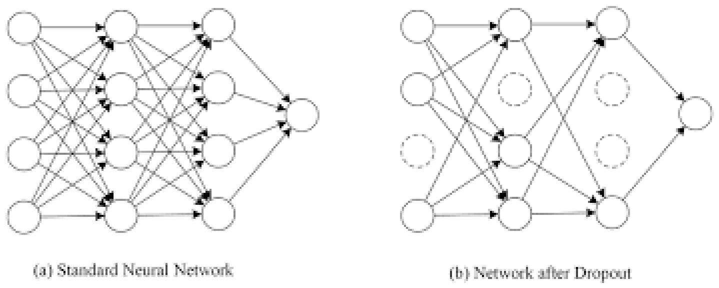
Ability to learn complex patterns: L2 > L1

- Because of the nonlinear nature of the L2 regularization, it is able to learn more complex patterns that the L1 regularization

Dropout

- Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel.
- During training, some number of layer outputs are randomly ignored or “dropped out”. This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different “view” of the configured layer.
- Because the outputs of a layer under dropout are randomly subsampled, it has the effect of reducing the capacity or thinning the network during training. A wider network with more nodes is thus required when using dropout

- Dropout simulates a sparse activation from a given layer, which in turn, encourages the network to actually learn a sparse representation as a side-effect.


(a) Standard Neural Network (b) Network after Dropout

Applying dropout in the forward and backward passes

- For the forward pass, we know that each neuron has a probability of being turned off by probability p. Therefore we can model dropout by scaling the output of every neuron to 0, with a probability p. It does not contribute any further during both forward and backward pass, which is essentially dropout.

- During the training phase, the network was trained with only a subset of the neurons. So during the testing phase, we have to scale the output activations by a factor of p. A simpler and commonly used alternative called *Inverted Dropout,* which scales the output activation during the training phase by 1/p so that we can leave the network untouched during the testing phase.

- In the backward pass, the dropout layer has no learnable parameters and doesn't change the size of the output. So, we simply backpropagate the gradients through the neurons that were not killed off during the forward pass

Data augmentation

- Data augmentation is a strategy that increases the diversity of data available for training models, without actually collecting new data, especially when using image data.

- Popular data augmentation techniques include cropping, padding, and horizontal flipping, scaling, shifting, etc.

Early stopping

- Early stopping is a regularization technique that relies on cross-validation.

- Cross-validation is a method of model validation that splits the data in different ways in order to obtain better estimates of "real world" model performance and minimize validation error.

- While model training, observe the performance of validation error. When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.

This concludes another topic in our ML cheat sheet series. I hope you found it helpful. Thank you for reading :)

Thanks to Emma Amor.

Machine Learning Data Science Deep Learning AI Regularization